UNIVERSITÀ DI PARMA
Dipartimento di Ingegneria e Architettura

Authenticity: Message Authentication and Digital Signature

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Course of Cybersecurity, 2022/2023

http://netsec.unipr.it/veltri

Message Authentication

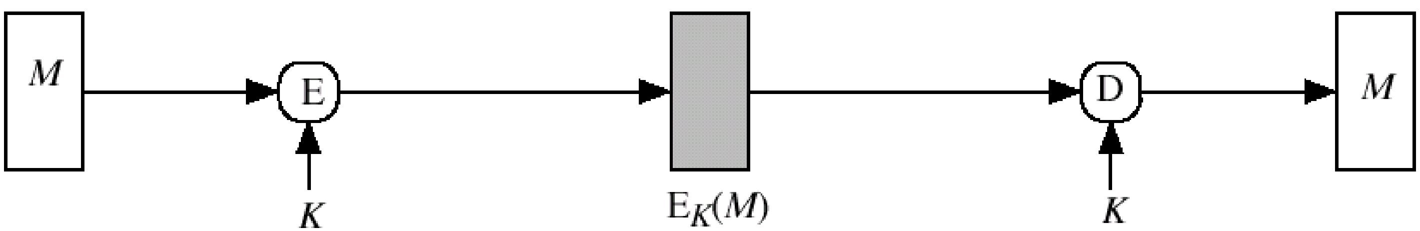
- Message authentication is concerned with:
  - **protecting the integrity of a message**
  - **origin authentication**
    - validating identity of originator
  - **in some cases, also non-repudiation of origin (dispute resolution)**

- Possible approaches:
  - **Symmetric mechanisms**
    - Symmetric encryption
      - sometimes together with an internal integrity check
    - Message Authentication Code
      - keyed one-way functions
  - **Asymmetric mechanisms**
    - Asymmetric encryption
    - Digital signature

Message Authentication using symmetric keys

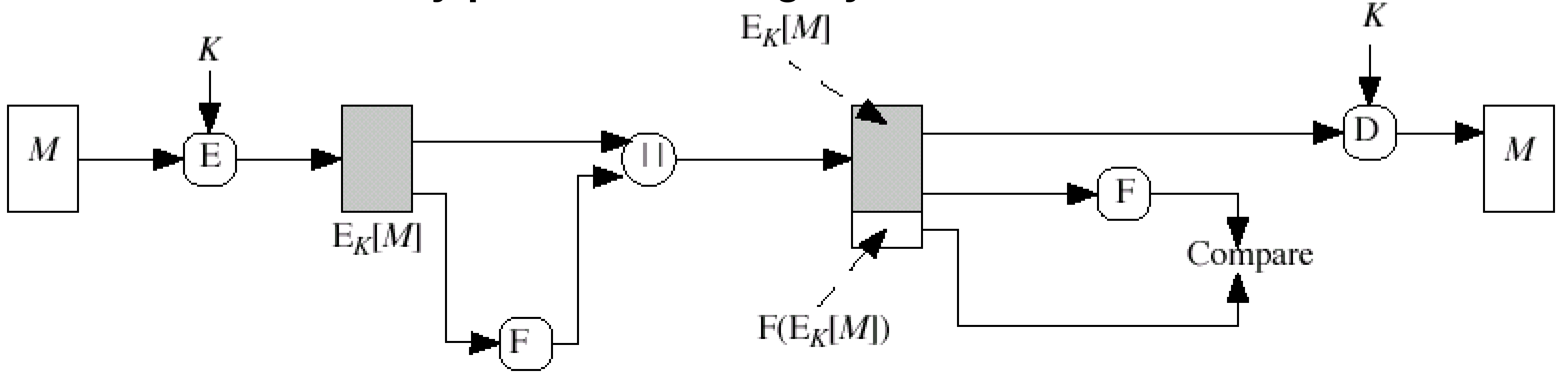
Msg. Auth. - Secret-key Encryption

- Symmetric encryption:
  - **encryption may provide both confidentiality and origin authentication**
  - **however, need to recognize corrupted messages**
    - based on the received message or with an explicit internal integrity check (see next slide)

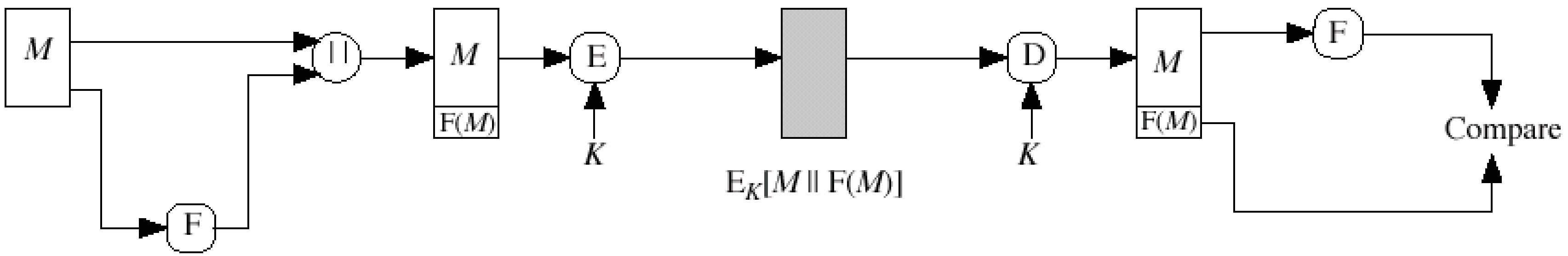


Msg. Auth. - Secret-key Encryption (cont.)

- External error control (checksum):
  - **does not securely protect the integrity**

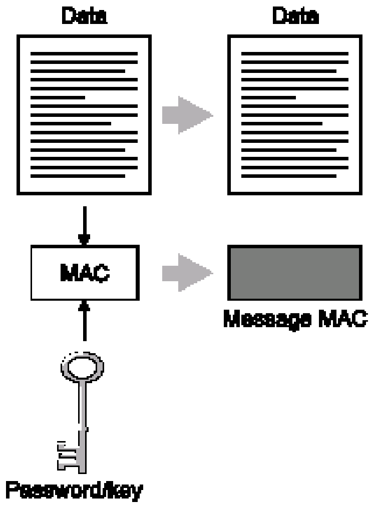


- Internal integrity check, through:
  - **a manipulation detection code (a sort of robust checksum)**



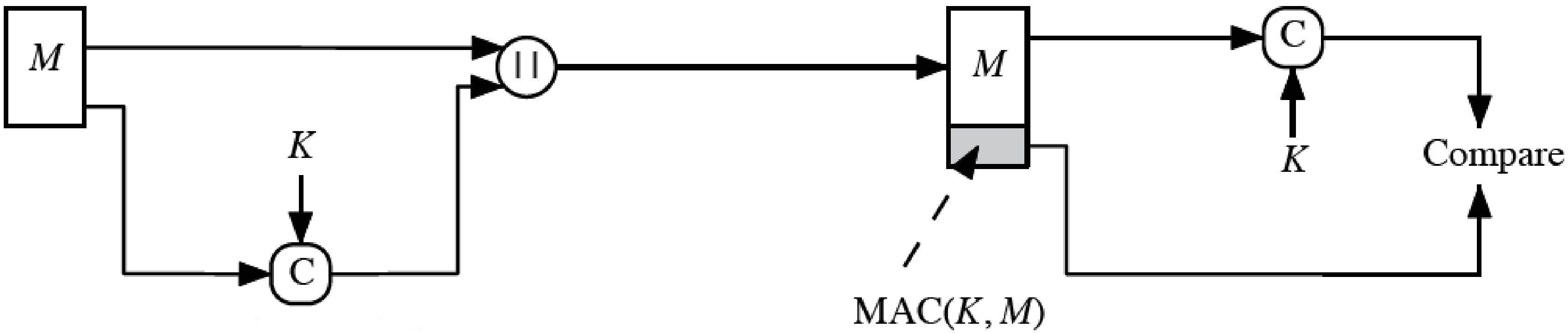
Message Authentication Code (MAC)

- Cryptographic checksum, generated by an algorithm that creates a small fixed-sized block
  - **depending on both message and a secret key K**
    - $\mathrm{MAC} = C_K(M) = C(K,M)$
  - **condenses a variable-length message M to a fixed-sized authenticator**
    - it doesn't need to be reversible
    - is a many-to-one function
      - potentially many messages have same MAC
      - but finding these needs to be very difficult



Message Authentication Code (MAC)

- Use of MAC for message authentication:
  - **appended to message as a signature**
  - **receiver performs same computation on message and checks it matches the MAC**
    - provides assurance that message is unaltered and comes from sender
  - **can be used also without enforcing confidentiality**



Message Authentication Code (MAC) (cont.)

- In case secrecy is also required
  - **use of encryption with separate key**
  - **can compute MAC either before or after encryption**
    - is generally regarded as better done before
- Why use a MAC?
  - **sometimes only authentication is needed**
  - **sometimes need authentication to persist longer than the encryption (eg. archival use)**
- MAC is similar but not equal to digital signature

Requirements for a MAC function

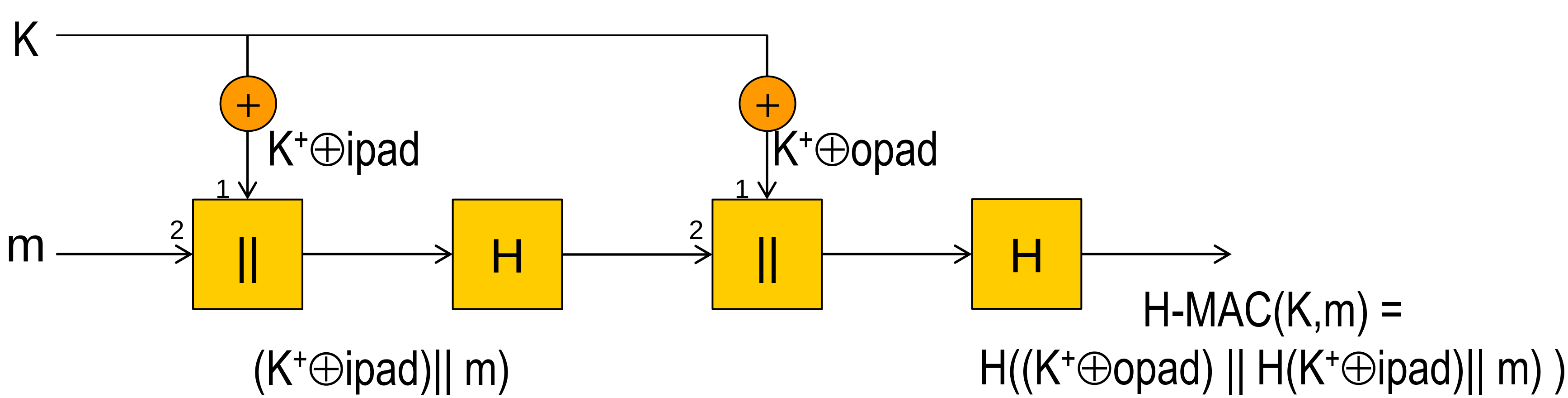
- MAC functions have to satisfy the following requirements:
  - **knowing a message and MAC, is infeasible to find another message with same MAC**
  - **is infeasible to find two messages with same MAC**
- Additional properties:
  - **MAC value should be uniformly distributed**
  - **MAC should depend equally on all bits of the message**
- Properties similar to hash functions
  - **in addition, MAC uses an input key**

Hash Message Authenication Code (H-MAC)

- Mechanism for message authentication using cryptographic hash functions in combination with a secret shared key
- Specified as Internet standard RFC2104
- HMAC can be used with any iterative cryptographic hash function, e.g., MD5, SHA-1, SHA-256, etc
  - **the cryptographic strength of H-MAC depends on the properties of the underlying hash function**

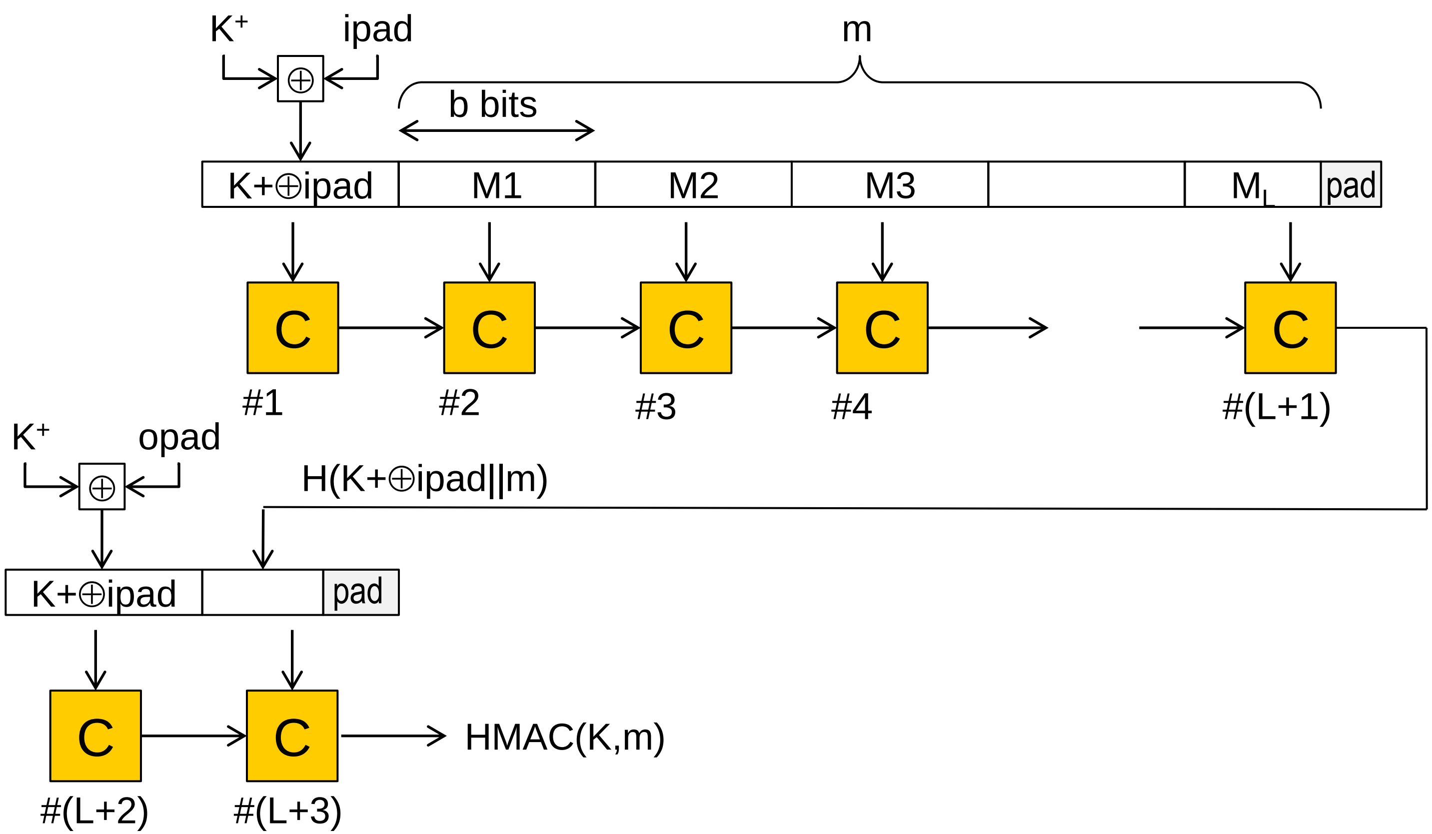
H-MAC (cont.)

- HMAC(K,m) = Hash[(K^+ XOR opad) || Hash[(K^+ XOR ipad)|| m)]]
  - **where K+ is the key 0-padded out to size b**
    - *b* is the size of the processing block
      - e.g. *b* = 512bits = 64bytes for SHA1
    - if *K* is longer than *b* bytes it is first hashed using H
  - **and opad, ipad are specified padding constants**
    - ipad = the byte 0x36 repeated *b/8* times
    - opad = the byte 0x5C repeated *b/8* times

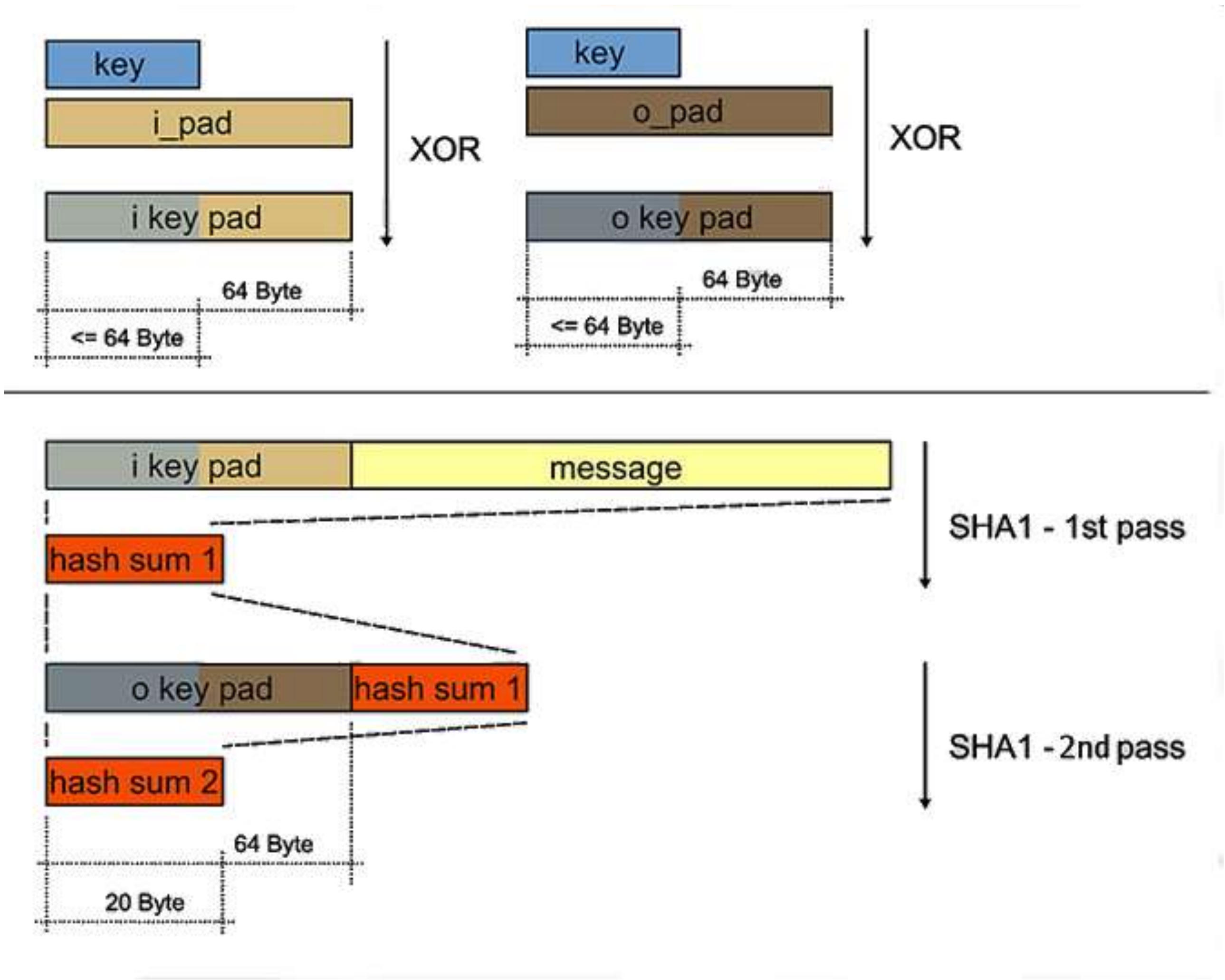


H-MAC (cont.)

- Overhead is just 3 more hash inner calculations (compression function *C*) than the message needs alone for computing message digest



Example - H-MAC-SHA1



Truncated H-MAC

- A well-known practice with MACs is to truncate the output of the MAC and output only part of the bits
  - **Advantage: less information on the hash result available to an attacker**
  - **Disadvantage: less bits to predict for the attacker**
- It is recommended to let the output length *t* be not less than half the length of the hash output and not less than 80 bits
- HMAC that uses a hash function *H* with *t* bits of output can be denoted as HMAC-*H*-*t*
  - **Example, HMAC-SHA1-80 denotes HMAC computed using the SHA-1 function and with the output truncated to 80 bits**

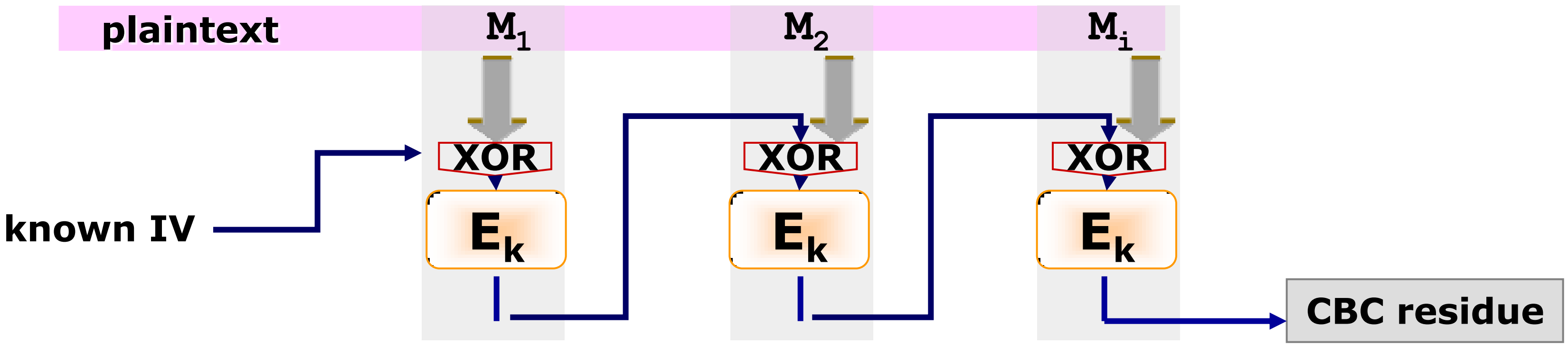
MAC Security

- Attacks:
  - **Cryptanalytic attacks**
  - **Brute-force attacks**

- Transient effect
  - **a published breaking of a MAC scheme would lead to the replacement of that scheme, but would have no adversarial effect on information authenticated in the past**
  - **this is in contrast with encryption, where information encrypted today may suffer from exposure in the future if, and when, the encryption algorithm is broken**

Using Symmetric Ciphers for MAC

- If a cryptographic algorithm is available with CBC mode, a way of generating a MAC is to compute the CBC but keep only the last block (named CBC residue) as MAC value



- CBC-MAC standard mode use IV=0
- E.g. Data Authentication Algorithm (DAA) (now obsolated) is a CBC-MAC based on DES
- Can use also other block cipher chaining modes and use final block

Using Symmetric Ciphers for MAC (cont.)

- Another approach could be to encrypt the hash
  - **$MAC_K(m) \equiv E_K(H(m))$**

- The main drawbacks of MAC based on symmetric ciphers are:
  - **the lower speed (e.g. compared with HMAC)**
  - **the size of the output that may be too small for security (it depends on the block cipher)**

Authenticated Encryption

Authenticated Encryption

- Sometimes both message authentication and encryption are required
- Authenticated encryption (AE) is a cryptographic system that simultaneously protects confidentiality and authenticity (integrity)
- There are four common approaches to providing both confidentiality and authenticity for a message m:
  - **Hash-then-Encrypt ***
    - $E_K(m||H(m))$
  - **MAC-then-Encrypt**
    - $E_{K2}(m|| MAC_{K1}(m))$
  - **Encrypt-then-MAC**
    - $C|| MAC_{K1}(C))$, where $C=E_{K2}(m)$
  - **Encrypt-and-MAC**
    - $E_{K2}(m) || MAC_{K1}(m)$
- Methods 2, 3, and 4 use two different keys

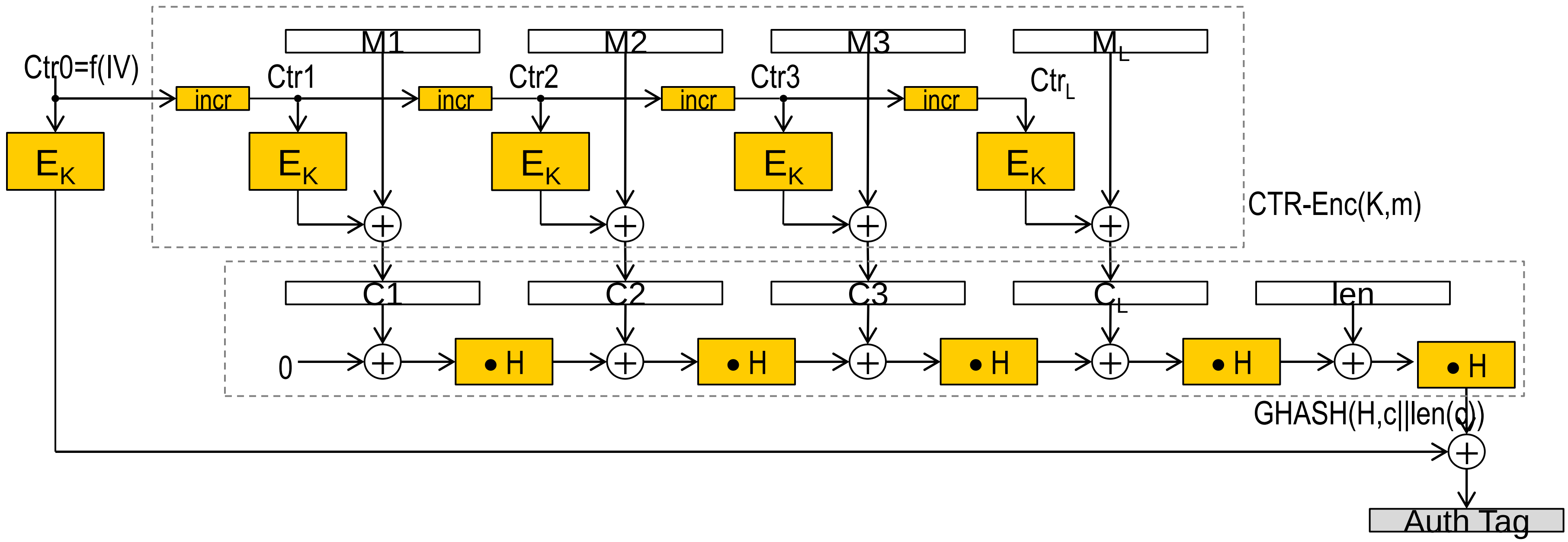
Authenticated Encryption (cont.)

- Example:
  - **CCM (Counter mode with CBC-MAC)**
    - Encrypt-and-MAC
    - ciphertext: c = CTR-Enc(K,m)
    - auth tag: t = Enc(K,Ctr0) ⊕ CBC-MAC(K,N||m)
      - » where *N* is a nonce value, *Ctr0* is the first generated counter (then *Ctr1*,*Ctr2*,etc are used for encrypting)

- A more efficient design of an AE system is to process the message only one time (just one pass)
  - **instead of separately encrypting and computing the MAC**
  - **e.g. Galois/Counter Mode (GCM)**

Example: Galois/Counter Mode (GCM)

- It is Encrypt-and-MAC:
  - **ciphertext: c= CTR-Enc(K,m), where Ctr0=f(IV)**
  - **auth tag: t= E(K,Ctr0)⊕GHASH(H,c||len(c))**
    - where GHASH(k,x) is a non-cryptographic keyed hash function
    - H= Enc(K,0)
    - X•Y is multiplication operation for the binary Galois field of 2^{128} elements
      - modular multiplication of binary polynomials of degree less than 128



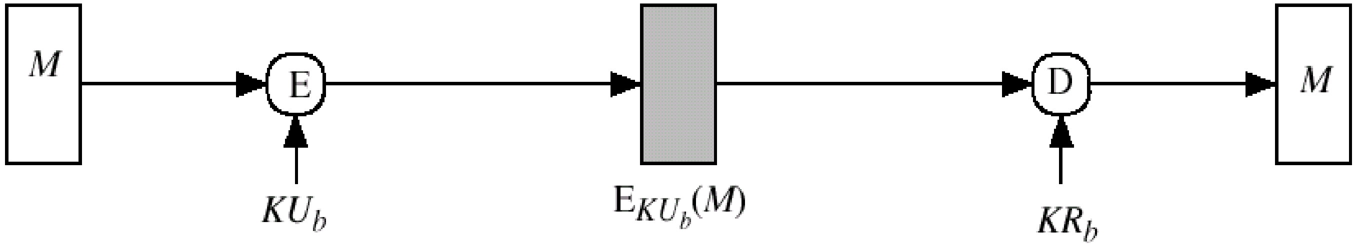
Authenticated Encryption with Associated Data

- Authenticated Encryption with Associated Data (AEAD)
  - **in addition to the plaintext '*m*' (that has to be AEed) there is extra data '*a*' that has to be only authenticated**
    - e.g. for protecting a network packet:
      - packet header (*A*) only authenticated since it must be readable,
      - packet payload (*m*) authenticated and encrypted
- Examples:
  - **CCM (Counter mode with CBC-MAC) AEAD**
    - c=CTR-Enc(K,m)
    - t=CBC-MAC(K,N||A||m) ⊕Enc(K,Ctr0)
  - **GCM (Galois/Counter Mode) AEAD**
    - c=CTR-Enc(K,m), where Ctr0=f(IV)
    - t=Enc(K,Ctr0)⊕GASH(H,A||c||len(A)||len(c)), with H=Enc(K,0)

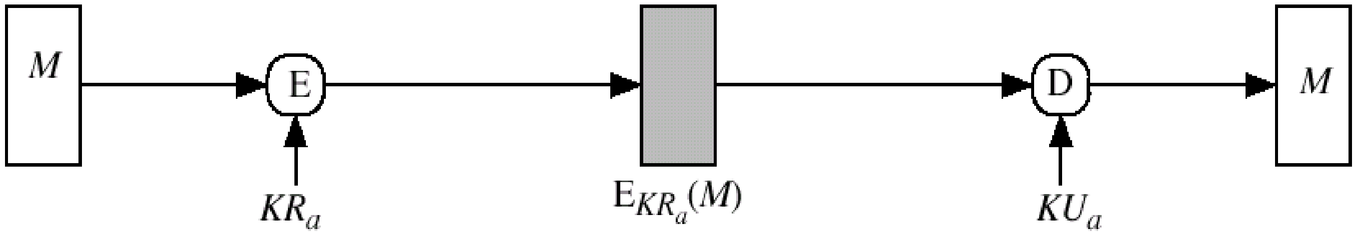
Digital Signature

Msg. Auth. - Asymmetric Encryption

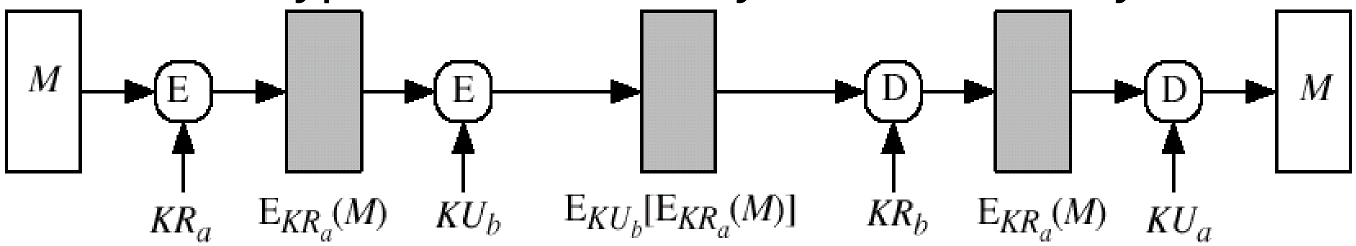
- Asymmetric encryption with public key: confidentiality



- Asymmetric encryption with private key: authentication
  - **however need to recognize corrupted messages**



- Asymmetric encryption with both keys: confidentiality + authentication



Msg. Auth. - Asymmetric Encryption (cont.)

- if public-key encryption is used
  - **encryption with public key provides no proof of sender (no sender authentication)**
    - since anyone potentially knows public-key
  - **encryption with private key provides authentication of the sender**
    - if it is possible to distinguish, after decryption, between a valid message and a random string of bits
  - **both secrecy and authentication if**
    - sender "signs" message using their private-key
    - then encrypts with recipients public key
- Problems
  - need to recognize corrupted messages
  - the signs has the same cost of public-key encryption of the entire message

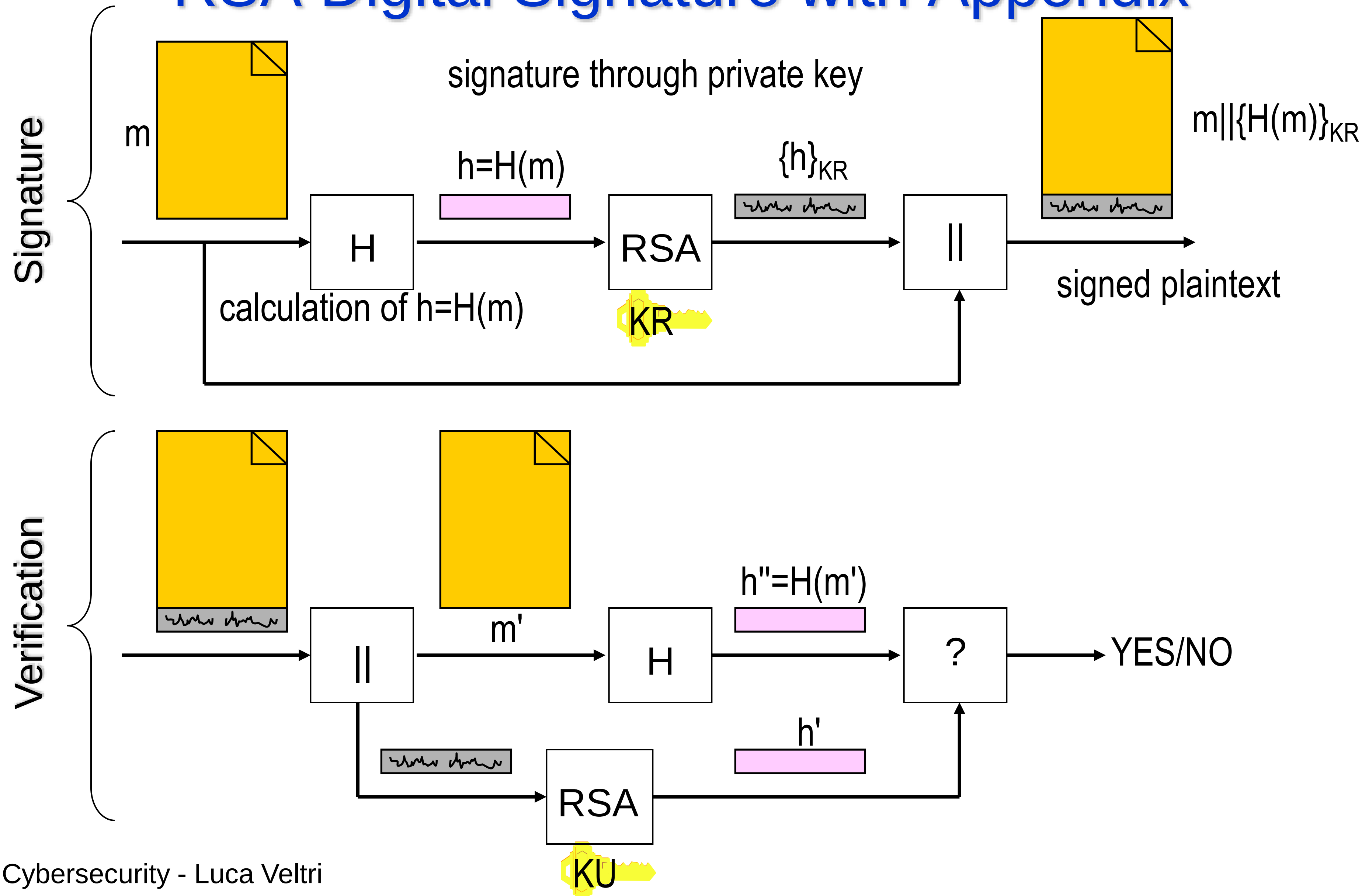
Digital Signature

- Digital Signature is an application in which
  - **a signer, say "Alice," "signs" a message *m* in such a way that**
  - **anyone can "verify" that the message was signed by no one other than Alice**
    - consequently that the message has not been modified
    - i.e. the message is a true and correct copy of the original
- The difference between digital signatures and conventional ones is that digital signatures can be mathematically verified
- A digital signature scheme (or mechanism) consists of
  - **a signature generation algorithm**
    - a method for producing a digital signature
  - **a signature verification algorithm**
    - a method for verifying that a digital signature is authentic (i.e., was indeed created by the specified entity)

Digital Signature (cont.)

- Two general classes of digital signature schemes:
  - **Digital signature schemes with appendix**
    - require the original message as input to the verification algorithm
  - **Digital signature schemes with message recovery**
    - do not require the original message as input to the verification algorithm
    - in this case, the original message is recovered from the signature itself
- A digital signature scheme is said to be:
  - **deterministic**
    - signing operation is a one message-to-signature transformation
  - **randomized**
    - if the signing operation is a function also of a second parameter, leading to an indexing set of message-to-signature transformations

RSA Digital Signature with Appendix



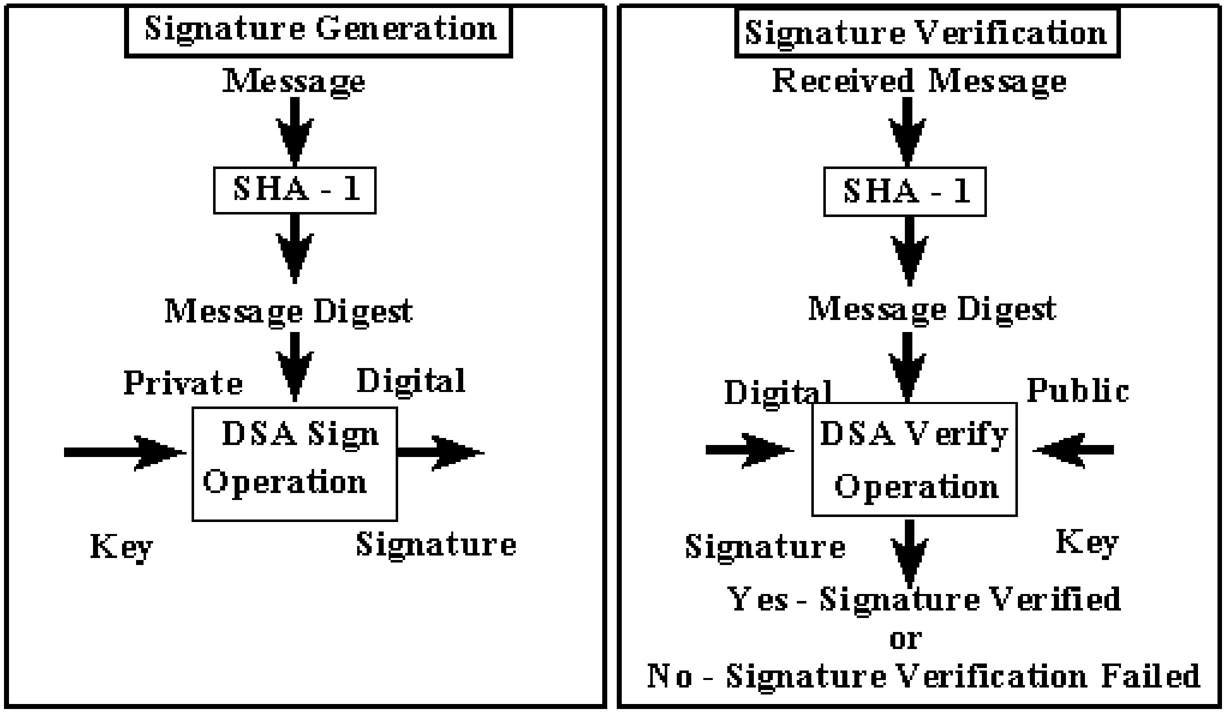
RSA Signature

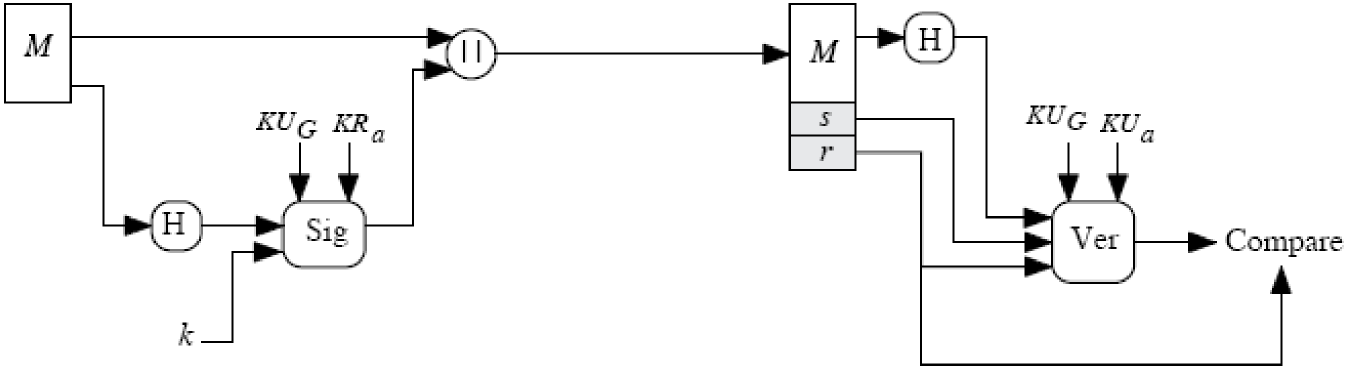
- PKCS#1 "RSA Cryptography Specifications Version 2.2" (RFC 8017) specifies two encoding methods for signatures with appendix:
  - **RSASSA-PKCS1-v1_5**
    - uses deterministic encoding
  - **RSASSA-PSS**
    - uses probabilistic encoding
      - includes a salt value
- These signature schemes combine signature/verification primitives with an encoding method for signatures
  - **a message encoding operation is applied to a message to produce an encoded message, which is then converted to an integer and processed by RSA signature primitive**
- Although no attacks are known against RSASSA-PKCS1-v1_5, RSASSA-PSS is preferred in new applications

Digital Signature Standard (DSS)

- DSS (Digital Signature Standard)
- Proposed by NIST (U.S. National Institute of Standards and Technology) & NSA in 1991
  - **FIPS 186**
- Based on an algorithm known as DSA (Digital Signature Algorithm)
  - **is a variant of the Elgamal (Taher Elgamal) scheme**
  - **uses hash algorithm, size *N* (e.g. SHA1, size 160)**
  - **uses *N*-bit exponents**
  - **creates a *2N* bit signature but with 1024 (or more) bit security**
- Security depends on difficulty of computing discrete logarithms

DSS Operations



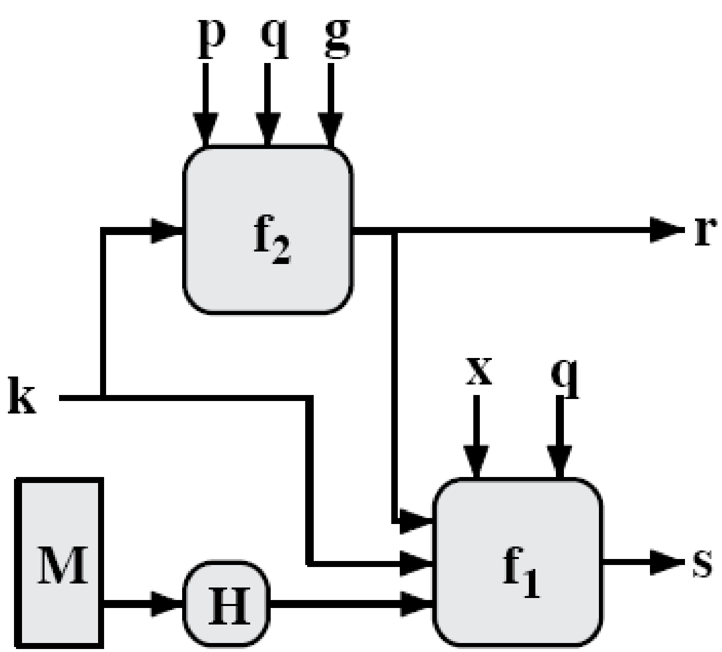


DSA Key Generation

- have shared global public key values (p,q,g)
  - ***L* and *N* are respectively the key length and hash length**
    - L = 1024 or more, and multiple of 64 (e.g. 1024, 2048, 3072,..)
    - N = 160 or more (e.g. 160, 256, ..)
  - **take a large (*L*-bit) prime p**
  - **choose q, a *N*-bit prime factor of p-1**
    - in practice, you can choose *q*, and then *p* such that *(p-1)* is multiple of *q*
  - **choose *g* such that its multiplicative order modulo *p* is *q***
    - in practice, $g=a^{(p-1)/q} \bmod p$
      - for some arbitrary *a* with 1<*a*<p-1, with $a^{(p-1)/q} \bmod p > 1$
- choose $x<q$
- compute $y = g^x \bmod p$
- public key = (p,q,g,y)
- private key = x

DSS Signing and Verifying schemes

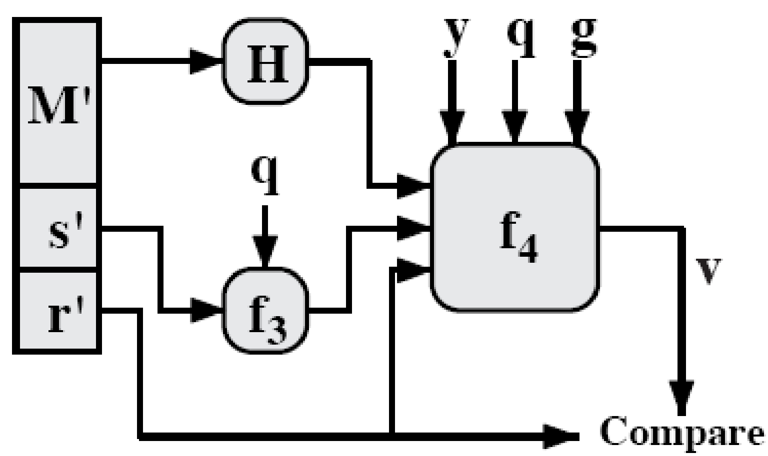
- Signing



$r = f2(k,p,q,g) = (g^k \bmod p) \bmod q$

$s = f1(H(m),k,x,r,q) = (k^{-1}(H(m)+xr)) \bmod q$

- Verifying



$w = f3(s,q) = s^{-1} \bmod q$

$v = f4(p,q,g,y,H(m),w,r) =$
$= ((g^{H(m)w \bmod q}\, y^{rw \bmod q}) \bmod p) \bmod q$

DSA Signature Creation

- to sign a message *m* the sender generates:
  - **a random signature key `k`, `k<q`**
    - N.B.: `k` must be random, be destroyed after use, and never be reused
- computes the message digest (e.g. SHA-1) of the message *m* :

  $\mathbf{h = H(m)}$

- then computes signature pair:

  $\mathbf{r = (g^k \bmod p) \bmod q}$

  $\mathbf{s = k^{-1}(h+x\cdot r) \bmod q}$

- sends signature `(r,s)` with message *m*

DSA Signature Verification

- having received m & signature (r,s)
- to verify a signature, recipient computes:

$$w = s^{-1} \bmod q$$
$$v = (g^{hw \bmod q}\, y^{rw \bmod q} \bmod p) \bmod q$$

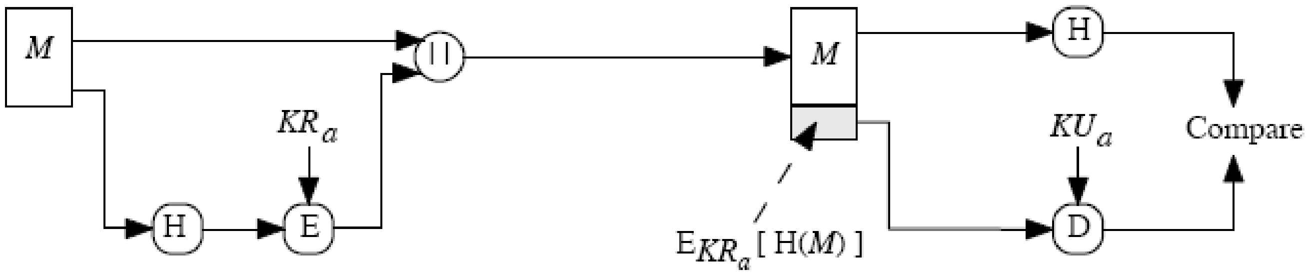
- if $v=r$ then signature is verified

- proof

$$\begin{aligned} v &= (g^{hw \bmod q}\, y^{rw \bmod q} \bmod p) \bmod q = \\ &= (g^{w(h+xr) \bmod q} \bmod p) \bmod q = \\ &= (g^{k} \bmod p) \bmod q = \\ &= r \end{aligned}$$

RSA vs. DSS signatures

- RSA signature



- DSS signature